Universidad del Valle de Guatemala
Programación de Microprocesadores
Sección No. 10
Docente: Juan Celada

Proyecto #3 de desarrollo: Diseño de programas con paralelismo en CUDA

Alejandro José Gomez Hernández 20347
Diego Alejandro Córdova Barrera 20212
Ana Paola de León Molina 20361

I. Índice

II. Introducción

Como parte del curso de programación de microprocesadores, se plantea el presente proyecto, con la finalidad de diseñar programas usando los conocimientos adquiridos en el curso. El enfoque del proyecto va respecto al paralelismo a nivel de threads en GPU. Es importante mencionar que la computación aplicada a GPU, en este caso CUDA, nace como una de las iniciativas de NVIDIA para brindar mayor rendimiento a programas en lenguajes como lo es C/C + +. Ahora, en cuanto a la motivación principal para hacer uso de CUDA en este proyecto nace de la necesidad de realizar programas con una capacidad de cómputo mayor a los que se compilan en una computadora ordinaria. En cuanto al sistema implementado para la recolección de datos, se hizo uso de una Raspberry Pi y distintos sensores, con un programa desarrollado en el lenguaje de Python. (Ver anexo no.1) Cabe destacar que, para ejecutar los códigos de CUDA se hizo uso de una instancia en el servidor virtual de la nube de Amazon Web Services (AWS).

III. Control de versiones

Para el desarrollo del presente proyecto, se hizo uso de la herramienta GitHub para llevar un control de versiones ordenado. Así mismo, fue de mucha utilidad para llevar el historial de los cambios que se llevaron a cabo durante la realización del mismo. Esta herramienta facilitó bastante el trabajo realizado debido a las distintas opciones que ofrece en cuanto al trabajo en distancia; adicional a las sesiones colaborativas en el software de Visual Studio Code para realizar la programación en tiempo real como grupo. El enlace para acceder al repositorio de GitHub se adjunta a continuación:
https://github.com/Nes15442/Proyecto-Final_Micro

IV. Funcionamiento del programa

Para el desarrollo del presente proyecto, se realizó la grabación de un video con la finalidad de demostrar su correcto funcionamiento, se adjunta a continuación:
https://youtu.be/oESEUqUbzm0

V. Cuerpo

i. Descripción de los sensores usados y sus características.

- Sensor DHT11:
    - Variables a medir con sensor:
        - Temperatura
        - Humedad
    - Características:
        - Al hablar del sensor DHT11, se puede mencionar que este es un sensor digital que mide temperatura y humedad relativa. La forma en la que funciona es usando un sensor capacitivo de humedad y un termistor para medir el aire, por medio de salida digital. Cabe destacar que este sensor es un poco menos preciso que los sensores más avanzados de esta serie (DHT22 y DHT21).

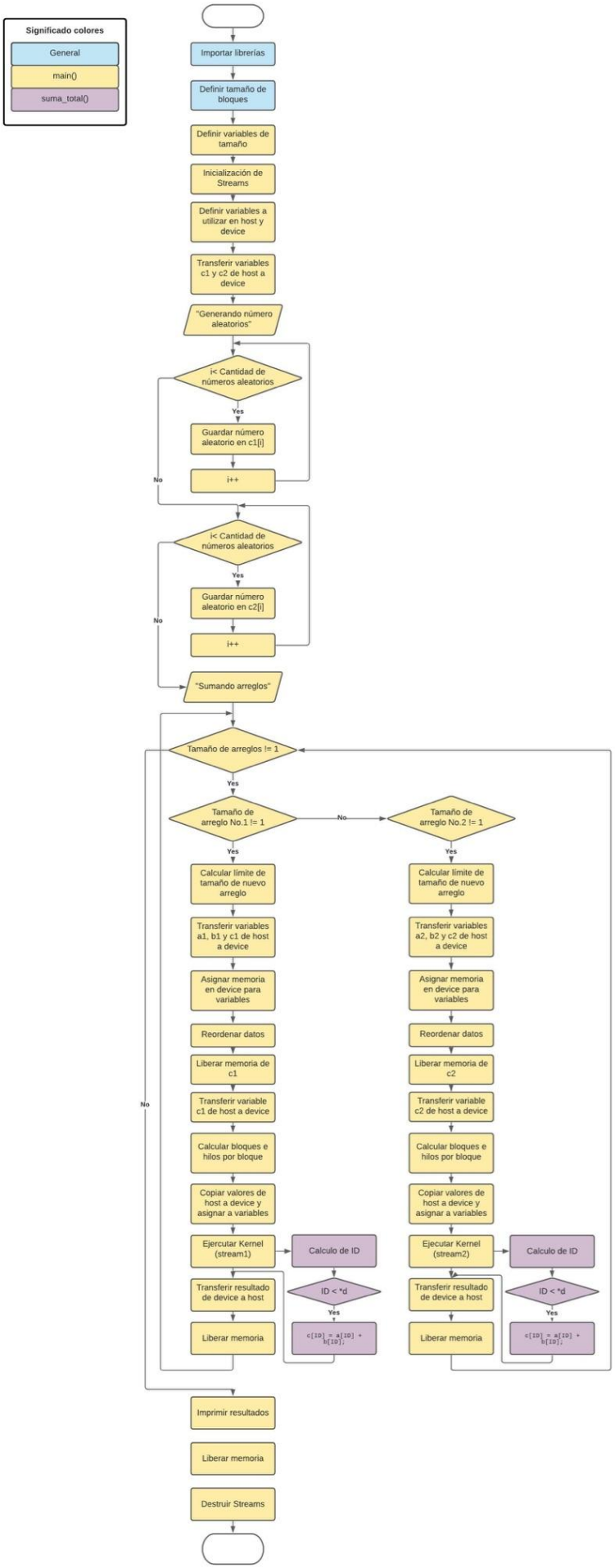
- Sensor BMP180:
    - Variables a medir con sensor:
        - Presión barométrica
        - Temperatura
    - Características:
        - En cuanto al sensor BMP180, se puede mencionar que este fue diseñado para brindar datos sobre la presión barométrica y temperatura. Cabe destacar que para su utilización dentro de la librería Adafruit en la sección de recolección de datos, se hizo uso de un módulo externo. Esto se usó debido a que Adafruit dejó de dar soporte al sensor.

- Sensor TL05i:
    - Variables a medir con sensor:
        - Temperatura
        - Humedad
    - Características:
        - Al hablar del sensor TL05i se puede mencionar que está basado en el sensor STH 31-D. Este, es un sensor con una interfaz I2C bastante eficaz en cuanto a la recolección de datos. Este sensor fue desarrollado en Tesla Lab de la Universidad Galileo. Cabe destacar que este sensor fue participante en el Encuentro Nacional de Robótica BALAM en 2021.

ii. Justificación del proyecto

En cuanto al proyecto realizado se puede mencionar que, respecto a la guía original se realizaron distintas modificaciones. Primero que todo, se descartó el uso de la información recopilada por los sensores debido a algunos de los conflictos ocurridos con el manejo de archivos JSON con CUDA. Por este motivo, se optó por utilizar listas con rangos distintos de números, para el análisis del efecto real respecto a la reducción del cálculo de medias aritméticas. Esto modificó bastante el enfoque del proyecto, sin embargo, los objetivos a cumplir se mantuvieron.

iii. Diagrama de flujo o UML describiendo el algoritmo del programa.


Significado colores
General
main()
suma_total()
Importar librerías
Definir tamaño de bloques
Definir variables de tamaño
Inicialización de Streams
Definir variables a utilizar en host y device
Transferir variables c1 y c2 de host a device
"Generando número aleatorios"
i< Cantidad de números aleatorios
Yes
Guardar número aleatorio en c1[i]
i++
No
i< Cantidad de números aleatorios
Yes
Guardar número aleatorio en c2[i]
i++
No
"Sumando arreglos"
Tamaño de arreglos != 1
Yes
Tamaño de arreglo No.1 != 1
No
Tamaño de arreglo No.2 != 1
Yes
Yes
Calcular límite de tamaño de nuevo arreglo
Transferir variables a1, b1 y c1 de host a device
Asignar memoria en device para variables
Reordenar datos
Liberar memoria de c1
Transferir variable c1 de host a device
Calcular bloques e hilos por bloque
Copiar valores de host a device y asignar a variables
Ejecutar Kernel (stream1)
Calculo de ID
ID < *d
Yes
c[ID] = a[ID] + b[ID];
Transferir resultado de device a host
Liberar memoria
Calcular límite de tamaño de nuevo arreglo
Transferir variables a2, b2 y c2 de host a device
Asignar memoria en device para variables
Reordenar datos
Liberar memoria de c2
Transferir variable c2 de host a device
Calcular bloques e hilos por bloque
Copiar valores de host a device y asignar a variables
Ejecutar Kernel (stream2)
Calculo de ID
ID < *d
Yes
c[ID] = a[ID] + b[ID];
Transferir resultado de device a host
Liberar memoria
No
Imprimir resultados
Liberar memoria
Destruir Streams

iv. Descripción de catálogo de las funciones desarrolladas para implementar el algoritmo de solución.

01. Suma_total(): Mediante esta función se realiza el cálculo correspondiente para conocer el identificador del elemento a analizar. Por lo mismo se realiza la verificación que el mismo se encuentre dentro del rango de elementos generales del programa y seguidamente se realiza la sumatoria en donde se almacena en la lista resultante c.
02. Main(): Esta función es la encargada de realizar todas las llamadas necesarias para el funcionamiento del programa, pues en este se conectan y enlazan todas las variables definidas en el host y en el device. Así también, se definen los Streams y se realiza la generación aleatoria de la información a utilizar.

v. Retos encontrados para la implementación y conclusiones sobre el proceso de implementación paralela con pthreads.

En primer lugar, en cuanto al mecanismo de captura y recolección de datos, se puede mencionar que, el problema inicial encontrado fue respecto a la conexión correcta de jumpers desde el sensor hacia la Raspberry Pi. Ahora, entre los retos que se encontraron para la implementación paralela, en primer lugar se encontró que el manejo de datos en formato json en el lenguaje de programación C + + se convirtió en un conflicto. Se hizo uso de varias librerías para el uso de JSON, con los datos del sensor, sin embargo, no se obtuvieron los resultados esperados. Por este motivo, se modificó el enfoque del proyecto, para que utilizara datos aleatorios en rangos predefinidos. Despreciando así, todos aquellos datos que se recolectaron en el plazo de tres días. Así mismo, otro de los retos encontrados respecto a la implementación de CUDA radica en la utilización de una instancia en AWS, debido a que se contó con ciertas restricciones de horario en los cuales era posible realizar la conexión.

VI. Conclusiones

- Se reforzaron los conceptos vistos en el curso respecto a la importancia de la utilización de la programación paralela en un ambiente de computación sobre tarjetas gráficas como lo es CUDA.

- Mediante la realización del programa se logró identificar los pasos necesarios para el correcto funcionamiento de un programa paralelo, aplicando los conocimientos de transferencia de datos de host a device y viceversa.

VII. Recomendaciones

- Para futuras réplicas de proyectos similares se recomienda contar con la instalación de las herramientas de CUDA para la ejecución local del código desarrollado. De esta forma, se evitará el uso de utilizar instancias en servicios como AWS.
- Así mismo, en cuanto al tipo de archivo a interpretar para el análisis de datos, se recomienda buscar alternativas al análisis de archivos JSON, debido a que puede llegar a ocasionar problemas con el manejo de librerías.

VIII. Bibliografía

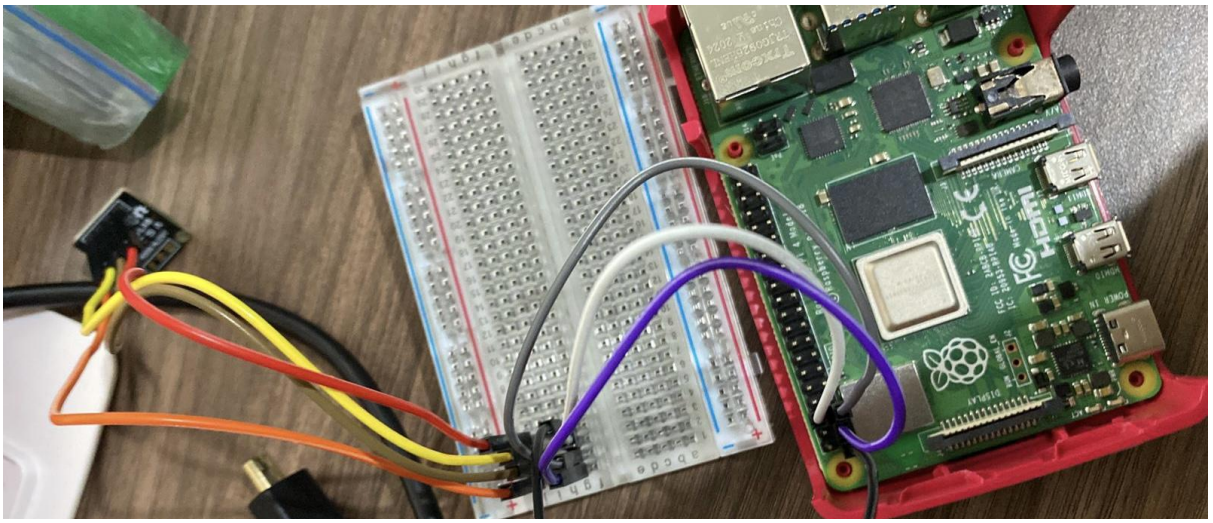
NVIDIA. (2021). Programming Guide :: CUDA Toolkit Documentation. (C) Copyright 2005. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Heller, M. (2018, 30 agosto). What is CUDA? Parallel programming for GPUs. InfoWorld. https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html

Run:AI. (2020, 16 octubre). CUDA Programming: An In-Depth Look. https://www.run.ai/guides/cuda-programming/

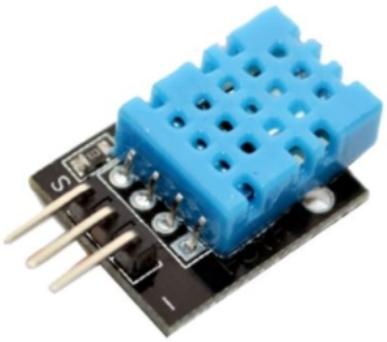
VIV. Anexos

Anexo No.1: Representación de la conexión de Raspberry Pi con sensores

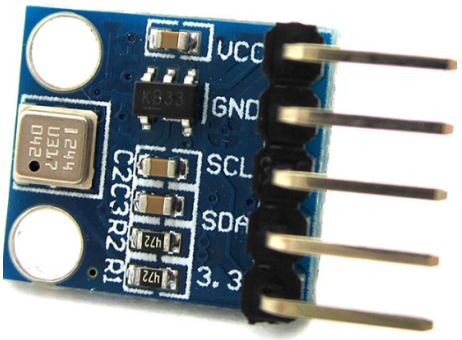


VV. Apéndice

Apéndice A. Sensor DHT11:



Apéndice B. Sensor BMP180:



Apéndice C. Sensor TL05i: